MASY: MAnagement of Secret keYs for Federated Mobile Wireless Sensor Networks

Jef Maerien, Sam Michiels, Christophe Huygens, Wouter Joosen
IBBT-DistriNet
Department of Computer Science
Katholieke Universiteit Leuven
B-3001, Leuven, Belgium
Email: firstname.lastname@cs.kuleuven.be

***Abstract*—Wireless Sensor Networks are becoming federated and mobile environments. These new capabilities pose a lot of new possibilities and challenges. One of these challenges is to create a secure environment to allow multiple trusted companies to share and merge their sensor network infrastructure. The most basic need for a secure environment is the deployment of key material. However, most current day research assumes pre-shared secrets between the sensor nodes of most, if not all, companies in a federation. These solutions are often not scalable nor mobile enough to meet realistic business requirements. Additionally, most key deployment protocols totally omit any connectivity with back-end infrastructure. This paper proposes a novel deployment protocol for the MAnagement of Secret keYs (MASY). MASY allows secure deployment of a key to a sensor node when it enters a previously unknown network. By off-loading the trust creation process to the resource-rich back-end infrastructure, the burden on the sensor nodes remains very limited.**

***Index Terms*—Security, Key Distribution, Wireless Sensor Network.**

I. Introduction

Wireless Sensor Networks (WSNs) can be used for a very wide variety of applications. A classic example is environmental monitoring [1], where a single party monitors a certain location with statically located sensor nodes. However, the field of WSNs is moving away from those mono-application, single party, statically located use cases which have been dominating the research of the last few years. New use cases such as container tracking [2] and personal health monitoring [3] create a much more dynamic environment. WSNs will be used for many, concurrent, and evolving applications, cooperating with many nodes from several different companies which work together as a federation and, most importantly, sensor nodes will be highly mobile [4]. These new assumptions pose a number of new challenges: how to manage the evolving applications on these resource constrained nodes, how to cope with the high mobility of the nodes and how to secure all the necessary relationships between the multiple and evolving companies and nodes that are present in a single WSN.

So, in many real world applications, WSNs consist out of many nodes from many different and changing companies (see figure 1). Additionally, the gateway connecting the nodes with the Internet is managed by another company, the local authority. All these nodes need to be able to offer and consume each others services securely. The most basic of such services is the ability to securely route all the messages over the network. How to set up the trust and key material required for the security in such a mobile and federated environment remains a significant challenge.

Current security research in WSN often makes the assumption that the necessary key material is either partly or entirely present on the sensor nodes at deployment time. In a mobile environment this assumption is not scalable. Additionally, any connectivity with the Internet or back-end infrastructure is omitted in this research. On the other hand, security research in the mobile and embedded area of Vehicle Area Networks (VANETs) focuses mostly on the use and deployment of asymmetric certificates, which is too heavy weight for most of the current generation of sensor nodes.

In this paper, we propose a protocol for the MAnagement of Secret keYs (MASY), which allows a node to build a trust relationship in the form of a shared symmetrical key with the local authority without pre-agreeing secrets. It requires that the company that hosts the gateway and the company that owns the sensor node trust each-other. However, MASY does not require to exchange any secrets before the node actually enters the network. Additionally, it only requires that the nodes support symmetric encryption.

The paper is structured as follows. Section II describes the context of this paper and lists the main requirements of the key deployment protocol. Section III gives an overview of the current related work in WSN and VANET security. Section IV lists the main assumptions our protocol makes and proposes the MASY protocol. Section V provides a security analysis and implementation overview. Section VI lists some interesting options for extending the protocol and summarizes the main contributions of this paper.

II. Context and Requirements

A. Use Case

In this paper, we apply the use case of container tracking to illustrate the need to identify multiple companies and to share services. Containers travel around the world, visiting many ports and warehouses along their journey. Many companies are interested in data related to the containers. The company that owns the containers wants a continuous report on the status of their containers. Because of this need, the company

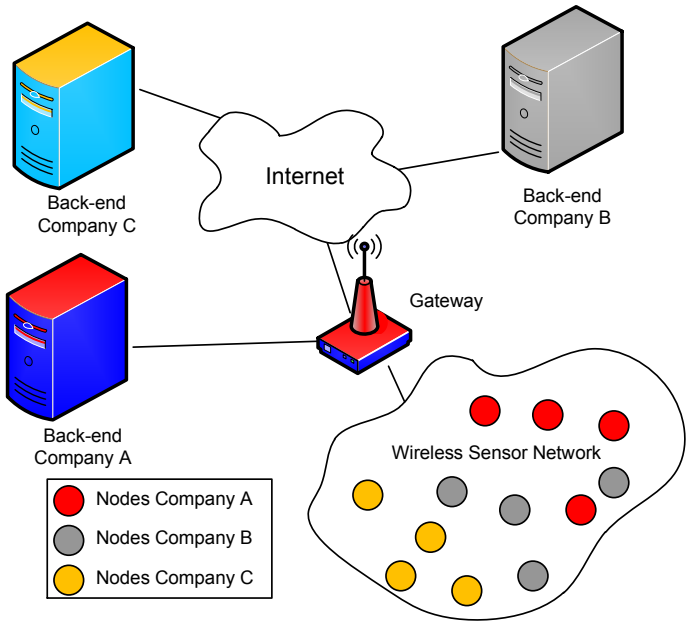

Fig. 1. Example of a Federated Wireless Sensor Network.

installs one or more sensor nodes to continuously monitor the temperature and humidity in its containers. Local authorities also want data from the containers for customs declarations. Other companies need to be able to communicate with the sensors in the containers to route their messages to the gateway or potentially use services of the sensor nodes.

The company that owns the containers wants to receive the monitoring feed from the sensors in the container. In order to allow this feed, the sensor nodes need to securely communicate with the back-end servers of the company. Of course the sensor nodes are able to encrypt the message to the back-end server. However, in order to securely communicate, the messages have to be securely routed to the gateway. If the routing in the network does not happen in a secure way, any malicious party can intercept all message and prevent the nodes from sending any information.

In a port or warehouse, the local authority (the company that owns the warehouse) also wants to monitor the assets that are present. One option is that the local authority deploys a sensor network of its own. This, however, would be costly, redundant and cannot measure the internal state of containers. An alternative is that the container owners allow the local authority to use some of the data of their sensors so the local authority can have a real-time and accurate view of all goods that are currently in its care.

The companies owning other containers in the vicinity of the original container also want information and services of their sensors. The most important service is the delivery of messages to the gateway using other nodes, so that the messages can be delivered to their own back-end servers. Additional services might be offered for other applications. For example a company could use sensor data to ensure fault tolerance of sensors in case of sensor node failure. Some sensors can also provide more accurate or different types of data than others. For example, the ambient temperature of containers holding chemical, perishable or pharmaceutical goods must be closely monitored. In order to limit the redundancy and cost for the companies, a federation can be formed and the data gathered from the sensors shared.

B. Sensor Network Challenges

In order to fulfill this scenario, four key requirements must be met: (1) nodes must be able to set up secure communications when they enter a previously unknown network, (2) the solution must be scalable, (3) code overhead must be limited and (4) communication overhead must be limited.

The first requirement relates to mobility. As containers and their nodes travel across the world, they cannot know the keys to all locations in advance. Since the nodes do not share any key material with the local network when they arrive, this key material must be deployed in an ad-hoc fashion. The sensor nodes however must know that the local key they receive is from a trusted source. If not, it might open the sensor nodes up to possible exploitation. On the other side, the local authority must be sure that it does not share the local key with an untrusted party, since this would lead to a breach of network security.

The second requirement is scalability. WSNs in ports and warehouses consist of hundreds if not thousands of nodes. Many nodes can enter or leave the network in a very limited time frame, for example when a container ship arrives in the port. This means the solution must be able to handle many nodes entering and leaving the network at any one time.

The third requirement is limited code overhead. Wireless sensor nodes in a commercial environment need to be very cheap. Memory and processing capabilities are very limited in these nodes, so applications need to be as small and light weight as possible.

The fourth requirement is limited communication overhead. Wireless sensor nodes are severely energy constrained. The amount of communication they can perform is limited. Once they reach that limit, they seize to operate until recharged. However, since containers are often far away from their owner, one cannot easily recharge the sensor nodes. So any communication overhead must be reduced to a minimum.

C. Attacker Model

In this paper we assume active external attackers that wish to remain undetected. This means that the attackers have the following capabilities:

- The attackers can monitor the entire network. However we assume that they cannot break any encrypted communication.
- The attackers can inject new messages into the network.
- The attackers can subvert nodes, revealing all their key material.

We do not consider denial of service or flooding attacks as they would clearly signal the presence of an attacker.

D. Security Requirements

A key deployment protocol has to meet several security requirements. We identify four security requirements: (1) confidentiality, (2) authentication, (3) survivability, and (4) availability.

The first security requirement is confidentiality. The key material has to remain confidential and can only be shared with trusted parties. When key confidentiality is breached, any attacker can gain access to the network and potentially start to exploit other sensor nodes. Given the mobile and ubiquitous nature of sensor nodes, it is very likely an attacker can obtain one or more sensor nodes and learn the confidential material through probing of memory. This means that other measures have to be available to recover from key disclosure.

The second security requirement is authentication. Nodes can only enter a network when a trust relation can be created between the sensor node and the gateway with guaranteed authenticity of both parties. So they have to authenticate each other. The company that hosts the network must authenticate the companies that want to enter the network so no malicious parties can join the network. On the other side the company that enters the network must ensure that the company that hosts the network can be trusted so it doesn't open up its sensor nodes to potential abuse.

The third security requirement is survivability. Survivability means that when one node or company is compromised, the remainder of the network can still recover from the breach or continue to operate without too much interference. In Wireless Sensor Networks, the nodes are fairly fragile and vulnerable. Malicious parties can fairly easily get access to these nodes since usually they are not in a secure location. Those malicious parties can then probe or corrupt those nodes so they gain access to the key material inside those nodes or make the nodes perform malicious operations. A key deployment protocol must be able to withstand or recover from node capture.

The fourth security requirement is availability. Since no new nodes can enter a network without having keys deployed to them, it is vital that the key deployment service is maximally available.

III. Related Work

This section evaluates related work. First the related work in WSN security research is discussed, then we look at the field of Vehicle Area Networks (VANETs), because VANETS have to deal with an inherent high mobility and connectivity for which several solutions have already been proposed.

WSN key deployment protocols can roughly be divided in two types: (1) symmetric key deployment protocols and (2) asymmetric key deployment protocols.

A. Symmetric Key Protocols

Symmetric key deployment protocols currently are the preferred protocols in WSNs. They have the advantage of being light weight in both communication overhead and code and execution overhead. Hence, it is currently considered more suitable for WSNs. Camtepe et al. [5] provide a survey of current key distribution protocols. At this time, there are two general categories of symmetric key protocols: (1) protocols with a pre-shared keys with the gateway and (2) protocols with a pre-distributed key ring.

The first category of symmetric key protocol are the protocols where every node shares a symmetric key with the gateway. The gateway acts as the Key Distribution Center in this scheme and can securely deploy a group key to each of the nodes. If two or more nodes want to securely communicate, they ask the gateway to generate a secret key for them and deploy it to these nodes. Examples of such a system are LEAP [6] and PAKA [7].

It is clear that this system is fairly scalable, secure and light weight. However, this system is not mobile. It is assumed that every node has a key pre-shared with the gateway. Since we assume that our nodes travel between multiple WSNs and are connected with multiple gateways, we cannot assume that they always share a key with the gateway.

The second category is the network-only category of key deployment. There is no central trust entity that is able to deploy new keys or key material. Every node has a pre-deployed key ring or some pre-deployed key material [8] which is used to generate new keys. When two or more sensor nodes want to communicate with each other, they compare the key material on their key ring and use the shared key material to generate a shared secret. If no keys are shared, a third party is searched with whom they both share key material. This trusted third party can then securely deploy a key to the first two nodes.

This is a very broad category with two sub categories, the probabilistic and the deterministic protocols. The probabilistic protocols have no guarantee that they have shared key material with a node nearby. There is a chance that a key is shared depending on the network size and the key ring size. These protocols have to search for a third party when no key is shared. The deterministic protocols guarantee that each node can securely create a key with each other node. However, in order to guarantee this, the network size must be fairly limited in order for each node to have unique key material. It is clear that these are light weight protocols. However, this system is not very scalable. To reliably share a key with each node in each container, the key ring would have to be quite large. If the node only shares a key ring with other local nodes, this system would not be mobile, since new nodes would have no way of securely receiving the key ring.

B. Asymmetric Key Protocols

The second type of key deployment protocols are the asymmetric key deployment protocols. It was assumed that asymmetric keys require too much communication overhead and processing cost to use in WSNs. Recently though, Elliptic Curve Cryptography has lowered the code and communication overhead required for asymmetric cryptography [9], making it a viable alternative for WSNs. Additionally, through the use of specialized encryption chips, the energy cost of encryption

can be significantly lowered [10]. However it does still pose some challenges.

To achieve mobility and to securely agree keys, the sensor nodes have to verify the gateway's certificate and the gateway must be able to verify the sensor nodes' certificates. Assuming many different Certification Authorities certify the gateways, the overhead on the sensor node would still be quite significant. These certificates also have to be kept consistent with Certificate Revocation Lists. These additional requirements to secure the key agreement protocol cause a significant communication overhead, which should be avoided on WSNs.

So while this solution might offer an advantage in mobility and scalability, the code and communications overhead are still significant. Furthermore currently most research on using asymmetric keys in WSNs is focused on evaluating and optimizing the implementations and usage of these protocols, while almost no research is done on the key deployment and key management of asymmetric keys in WSNs.

All of the related work mentioned here totally omits the fact that almost all sensor networks are connected to the Internet and have some form of back-end infrastructure available. This infrastructure can perform the resource intensive operations such as certificate validation and key agreement.

C. VANET Key Protocols

VANETs are ad-hoc networks of vehicles and roadside infrastructure. This unique field of wireless networks poses a great deal of challenges. The high mobility is of course one of the major challenges. Since in the future many critical applications will require Vehicle To Vehicle (V2V) communication, it is important that the communication between vehicles is secured. Securing these vehicle networks has recently caught the attention of several researchers. New and innovative key management schemes have been proposed to meet these challenges.

A common concept in VANET security is that each vehicle has two certificates: a permanent, global certificate and a temporary, local certificate. An example of such a protocol is the TACK protocol suggested by Struder et al. [11]. The global Certification Authority is the vehicle registration authority of the country where the vehicle is registered. This CA certifies an asymmetric key pair to each vehicle it registers. However, this certificate cannot be used in a local setting due to privacy constraints. So, in order to be able to securely communicate with other local cars, the local authority provides a service that provides each vehicle on their territory with a local certificate.

When a vehicle enters a new local network, it requests a local certificate. The vehicle provides his global certificate, which is validated by the local authority. The local authority also provides his certificate which is validated by the vehicle. When mutual trust is established, the local authority can deploy a new temporary local certificate. This certificate can then be used by the vehicle to set up secure communications with all vehicles in the territory of the local authority.

It is clear that this is a solution which meets the mobility requirement. Keys are deployed securely and in an ad-hoc fashion. However, it requires a lot of asymmetric key management and communication between the vehicles, the local and the global Certification Authority. The required communication and processing overhead makes this protocol unsuited for use in Wireless Sensor Networks.

To conclude, current security research in WSNs lacks mobility, while current mobile security (VANET) research is too resource-intensive to be used in Wireless Sensor Networks. Additionally, the pervasive presence of the Internet remains unused in current WSN security research.

IV. MASY Key Deployment Protocol

A. Assumptions

MASY makes the following assumptions: (1) nodes can only perform symmetric cryptography operations, (2) each node shares a symmetric key with the back-end of its company, (3) when a node enters a network it does not share any key material with the local network, and (4) a trust relationship exists between the owning company and the local authority that manages the gateway.

The first assumption is that nodes can only perform symmetric encryption. Recent research has shown that nodes are capable of performing asymmetric cryptography, however there still is a significant energy cost and communication overhead. That is why the protocol does not use asymmetric cryptography on the sensor nodes themselves.

The second assumption is that each node shares a symmetric key with the company. A trust relationship in the form of a secret key is needed in order to securely communicate with each other. Since all sensor nodes undoubtedly start their journey at the location of the company, it is reasonable to assume that the company can securely deploy a secret key onto the sensor nodes.

The third assumption states that a node does not share a key with the local network when it enters the network. When a node enters a network, MASY assumes it does not know anything about that network, so it cannot share a key with it. The alternative is that all the necessary keys are pre-loaded on the sensor node. Since we assume that sensor nodes travel between many networks, this would mean that it needs a significant amount of keys pre-loaded on the sensor node, which requires a significant amount of memory. If any key is compromised, an attacker can potentially abuse the sensor node. If the group keys are present on the gateway, the gateway cannot revoke its key since it is already present on the sensor nodes. If the company wants to revoke or change the key, it needs to be possible to remotely manage the key store of the sensor nodes while they are in the field, which is what we are attempting to achieve in this paper. Additionally, when nodes cannot receive new keys in the field, all the networks need to be known and reserved in advance.

The fourth and last assumption is that the company of the sensor node and the local authority who manages the network are in a mutual federation, so a trust relationship exists between them. If there is no trust between these two companies, a node should not be able to enter the network.

Term	Explanation
N	Nonce
CompIP	IP address of company back-end
M	MAC Address of the new node
GW	Gateway
BE	Back-end
NN	New node that enters the network
RN	Relay node
GK	Group key of the local network
CK	Company key
SK	Session key between GW and BE

TABLE I
LEGEND OF ABBREVIATIONS

Communication	Content	Size(bytes)
$NN \rightarrow RN$	hello: M,CompIP,N,$\{$M,CompIP,N$\}_{CK}$	32
$RN \rightarrow GW$	$\{$ hello $\}_{GK}$	32
$GW \rightarrow BE$	$\{$ hello , GK $\}_{SK}$	48
$BE \rightarrow GW$	$\{$ reply : M , $\{$GK$\}_{CK}$ $\}_{SK}$	24
$GW \rightarrow RN$	$\{$ reply $\}_{GK}$	24
$RN \rightarrow NN$	reply	24

TABLE II
PROTOCOL FOR DEPLOYING THE GROUP KEY ONTO A NEW NODE

The trust between these two parties can be verified by verifying each others certificates. By using a certificate based approach and a PKI, the gateway and company only need to trust a number of Certificate Authorities (CAs). If the parties are trusted by the CAs, then they can trust each other.

B. MASY Protocol

In this section we propose the MASY MAnagement of Secret keYs protocol for when a new node enters a network. The MASY protocol operates as follows (see fig 2 and tables I and II).

Step 1: NN enters an unknown network: When a sensor node detects it has entered a new network (for example because it can overhear encrypted communication or a gateway beacon), it sends out a hello message. This hello message consists out of the identity of the sensor node (a MAC address M), the IP address of the company back-end (CompIP), a counter nonce and a signature of this message with the key the node shared with the company. The identity of the sensor node is needed because the local network has to be able to somehow identify the sensor node and the back-end needs to be able to determine which node has entered the network. The gateway has to know the IP-address of the back-end in order to set up communications with it. The counter nonce is needed to guarantee freshness of the messages. This can also be a time-stamp if the sensor nodes are time synchronized with the back-end. The message has to be signed with the company key so the company can verify it was indeed a trusted sensor node that sent the hello message.

Step 2: RN node relays hello message: When a sensor node in the network receives a hello message, it must forward it toward the gateway. Each sensor node in the network knows who the gateway is and what route it needs to take. If several nodes receive the hello message, only one needs to send it. Since the relay nodes are already in the network, this traffic can be encrypted with the group key so only if an outsider is in communication range of the new sensor node, it can hear that a new sensor node wants to join.

Step 3: GW receives hello and relays it together with GK to BE: When the gateway receives the hello message, it parses it and determines who the back-end of the sensor node is. The gateway can see who the back-end is based on the IP-address of the gateway in the hello message.

Once the gateway knows the identity of the company, it can contact the company. Once a connection is made, the two parties can verify each other. This can be done by starting an SSL connection with mutual authentication. Both the sender and the receiver need to prove that they can be trusted by providing a certificate that can be verified. Once the certificates have been verified, a cipher suite and secret key (SK) are agreed upon and a secure channel is set up.

Once a secure channel is established between the gateway and the company back-end, the gateway can transmit the hello message and the network group key (GK).

Step 4: Transmitting the reply: Once the company receives the hello message, it can verify that the hello message originated from a node of the company by verifying the signature. If the signature is valid, the back-end can be sure that a sensor node is trying to connect with the gateway. When it then receives the group key over the secured channel, it can encrypt this group key with the company key and send it back to the gateway with the identification of the node M.

Step 5: Returning the reply to the NN: The gateway can then send the group key encrypted with the company key back to the new sensor node. This can be done by transmitting the reply to the sensor node that relayed the hello message. The relay node can then directly transmit the reply to the sensor node.

Step 6: NN receives GK: Once the sensor node receives the reply, it can decrypt the group key with its own company key. Once it has decrypted the group key, it can securely communicate with the network and gateway.

The group key has been deployed to the sensor node, without any prior trust between the sensor node and the gateway. The group key was always transmitted in an encrypted format so outsiders are not able to intercept it without compromising either the company key or the SSL channel between the gateway and the company back-end.

Once the group key is established the sensor node is a part of the network. It can securely communicate with its neighbours so it can set up secure routing tables and secure communications with each node in the network.

An alternative is that instead of deploying a group key to the sensor node, a unique key is deployed to the sensor node. This key would only be shared between the sensor node and the gateway. Once this key is deployed, the sensor node can securely communicate with the gateway. To communicate with other sensor nodes, the gateway can act as a Key Distribution Center. The sensor node can ask the gateway to generate a key

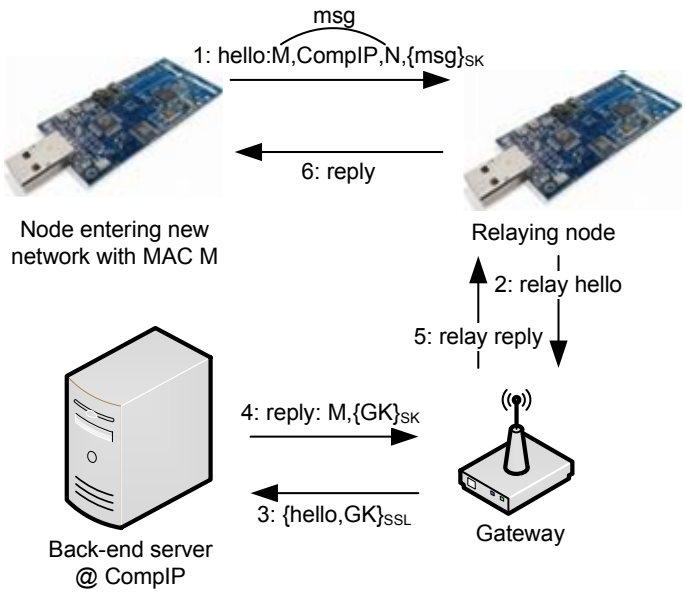

Fig. 2. Overview of the key deployment protocol.

to communicate with other sensor nodes, and the gateway can then deploy this key to the other nodes.

V. Analysis

A. Security Analysis

This section provides a security analysis of the MASY protocol. We look at the confidentiality, authentication, survivability and availability of MASY.

1) Confidentiality: MASY makes sure that the group key remains confidential at all times. MASY guarantees this since the group key is transferred from the gateway to the back-end server encrypted with an agreed upon session key over a secure SSL connection. The key is then encrypted with the company key that the company shares with the sensor node. An attacker thus cannot overhear the network key and start decrypting the traffic.

2) Gateway Authentication: A node might sometimes enter a malicious network with a malicious gateway. In such a case, it should not be able to initiate communications with the network. When it enters the network, the sensor node will still send out a hello message, since it has no way to decide whether it is in a trustworthy or malicious network.

If we assume that the gateway wants to start communication with the node, in order to potentially gain access to restricted services, it will send the hello message to the node's back-end server in order to deploy a key to the node. However, the gateway will fail to authenticate itself with the back-end server. Thus the sensor node will not receive the group key and will not enter into the network.

3) Company Authentication: An attacker might try to enter the network by acting as a new company. When the attacker's node enters the network, it will send out a hello message. This message will be sent to the gateway since the nodes in the network have no way to decide whether or not to trust such a node.

When the gateway receives the message it will try to connect with the back-end of the company. The gateway will then fail to authenticate the back-end server. Since the authentication fails, the gateway will not hand over the group key to this company, so the network key will remain secure.

4) Survivability: A WSN must survive the capture of one or more sensor nodes. Since the sensor nodes are very often far away from their owner, it is possible that an attacker captures a sensor node and manages to extract all secret information from this sensor node. This reveals all the keys that are present on the sensor node to the attacker.

If the node is connected to a network, the current group key is compromised. The first step to mitigate the attack is to identify the malicious node. Several mechanics have been proposed to identify malicious nodes when a node has been compromised, going from network clustering to intrusion detection systems and tamper evident hardware. However, we consider this out of scope of this paper.

Once the offending node is identified, the gateway can initiate a re-keying. Every sensor node which is still intact can receive a new group key in a secure way by repeating the protocol. The new group key is deployed using the company key which is unique for the nodes of each company. The compromised nodes will be excluded from the network.

5) Availability: The MASY protocol can fail if one of several services is not available.

The first service that has to be available is that the node has to be in range of a networked node or the gateway. It is clear that in order to be part of any network, the node must be able to send messages to the network. If this condition is not met, the node cannot enter the network.

The second service is secure routing to the gateway. If the node is part of an disconnected network without access to a gateway or the Internet, the protocol can not operate. However, if there is no Internet connectivity, the node cannot authenticate the network or the nodes in the network, so it should not be able to connect with the network.

The third service is the gateway. The gateway must be available in order to process the hello message and securely transmit the group key to the company. It could be that a gateway goes off-line due to an attack or accident. This threat can be mitigated by duplicating the gateway. The only knowledge a gateway must posses is the group key and the ability to authenticate foreign companies. By duplicating the gateway, more requests can be processed at the same time and the failure of one gateway would not interfere with the ability of new nodes to enter the network.

The last service that has to be available is the back-end service of the company. If this service is not available, the gateway cannot securely hand over the key to the company, nor can the company authenticate the gateway. Again this threat can be mitigated by duplicating the back-end servers. The server only has to know the company key that the node holds and has to have the ability to authenticate the gateway.

Implementation	ROM Overhead (kB)	RAM Overhead (B)
Tmote Sky (Contiki)	5,7	300
Sunspot	10,0	7000
LEAP protocol	17,9	600

TABLE III
IMPLEMENTATION OVERHEAD FOR MASY

B. Implementation

In this section we evaluate an implementation of MASY for 2 sensor platforms : the resource constrained Tmote Sky platform and the resource rich SunSPOT platform. We look at the application overhead and runtime memory for both programs (see table III) . Finally we evaluate the message size (see table II).

We have implemented the MASY protocol on the Tmote Sky sensor node platform running the Contiki Operating System. The Tmote Sky is a very limited sensor node with 1 mB storage, 48 kB flash memory and only 10 kB RAM. It is a very limited platform which is likely not able to perform asymmetric encryption. Contiki was selected as operating system because of its limited requirements and modular design.

The node application has a total size of 28880 bytes. This includes all the necessary Contiki packages and the AES encryption package. The standard Contiki platform with standard Rime communication support requires 23232 bytes. We used a non-optimised open source implementation of the AES algorithm which requires 4720 bytes. The protocol itself only requires another 928 bytes. By comparison, the LEAP protocol requires 17.9kB of ROM. The reason for the difference is that LEAP performs key establishment and key generation on the sensor node itself. It requires a much larger code base to function compared to our protocol.

The runtime overhead remains very limited at ca. 300 bytes which is mostly used for communication and encryption buffer. By comparison, the LEAP protocol requires a minimum RAM of 600 bytes when it has 1 neighbour and an additional 34 bytes for each additional neighbour.

We have also implemented the MASY protocol on the more powerful SunSPOT sensor node platform running the Squawk virtual machine. The SunSPOT has 512kB RAM and 4mB flash memory. We chose to also implement to protocol on this platform to show the platform independence of the protocol.

The SunSPOT suite has a size of 68528 bytes. This is quite a bit larger as the Contiki implementation. The actual protocol implementation overhead is estimated at about 7kB. The runtime overhead is estimated 10kB. Both the protocol implementation and runtime overhead are significantly larger compared to the Contiki implementation. The reason for this is (1) the inherent larger byte code and runtime overhead of Java applications, (2) the more object oriented implementation of the protocol and (3) when using any cryptography on the SunSPOT, all the cryptography libraries are included into the suite, many of which unnecessary. Further optimization could significantly lower the cost of this protocol on SunSPOT.

The relay functionality is currently not implemented on either platform. We made a gateway for each platform in Java. Two separate gateways are needed since currently the Contiki network layer is incompatible with the SunSPOT network layer. A single company back-end is implemented in Java for both applications. The choice for Java was made because it allows for rapid prototyping.

Two types of messages are transmitted through the sensor networks: hello messages and reply messages.

The hello message contains the node identification (8 bytes), the node home IPv4 address (4 bytes) a random nonce (4 bytes) and the signature (16 bytes). The signature is the message encrypted with the company key using AES encryptions with a key size of 16 bytes. This gives a total message size of 32 bytes, which is small and acceptable. The node that relays the message may need to add additional headers for encryption and the address of the gateway, but this is limited to another few bytes.

The reply message contains the node identification (8 bytes) and the encrypted group key (16 bytes). This gives a total message size of 24 bytes, again small.

The PAKA protocol uses a message from node to gateway which is 96 bits (12 bytes) large. The reply message is 64 bits (8 bytes) large. However it uses a smaller node identification (4 bytes) and smaller keys (8 bytes vs our 16 bytes). The TACK protocol on the other hand has a request size of 256 bytes and a reply size of 56 bytes. We note however that TACK uses asymmetric elliptic curve cryptography.

The computational cost of the protocol is low. Very little key management computations are done on the sensor node. The only significant computational work to be done on the sensor node for this protocol is one encryption operation when creating a hello message and one decryption operation when receiving the reply message.

We have not done measurements on network delay or total network overhead. The network delay depends on many factors such as number of nodes, speed of nodes, link quality, distance to gateway, etc. The protocol does not put limits on the delay, the reply just needs to reach the node. A timer on the new node could be needed however to ensure that even if the hello message is lost, a new message is sent.

To conclude, MASY has a limited communications overhead of a few dozen bytes. The MASY implementation overhead is limited in the order of a few hundred bytes, used mostly as memory space for keys, message buffers and communication buffers. Also, the AES implementation is not optimized for use in WSNs. Further optimization of the code of MASY and cryptography protocols can reduce the memory overhead. The computational overhead of MASY is limited since the cost of MASY is only one symmetrical encryption and one symmetrical decryption operation.

VI. FUTURE WORK AND CONCLUSION

A. Future Work

Currently we envision three options for further work: (1) optimizing message overhead for clusters of nodes travelling

together, (2) a re-keying mechanism and (3) extending the mechanism to deploy asymmetric certificates.

The first extension to MASY is the optimization of the protocol when nodes travel in clusters instead of alone. It is likely that a container is outfitted with multiple sensors that all want to be connected with the network, or that several containers of the same company travel together. In this case only one message should be sent for the group of sensor nodes. A single node, the group leader, is responsible for detecting when a new network is present. It then sends out the hello message and waits for the reply. When the reply is received, the group leader distributes the key over the sensor nodes.

The second extension is to provide a secure re-keying mechanism while excluding certain nodes. If a key has been compromised, it should be possible to restore the situation. Since the key is compromised, the new key cannot be distributed with the old key. However, by restarting MASY for each node (or cluster), each non-compromised node can receive the new group key in a secure fashion. This allows a compromised network to be restored to a secure state while excluding the compromising nodes/companies.

The third extension is to deploy certificates instead of symmetric keys. Assuming sensor nodes grow more powerful and new encryption techniques become available, it will become possible for sensor nodes to perform asymmetric encryption. In order to set up secure communications, the gateway must be able to deploy its certificate onto the sensor nodes. A PKI infrastructure could be used, where the sensor nodes can verify the certificate of the gateway. However, to limit the key store on the sensor nodes and remove the need to send and verify certificate chains, a system similar to MASY could be created, where instead of having a certificate signed by one of many possible CA's, the gateway could ask the owning company to provide it with a certificate signed by the company. This way the sensor nodes would only need the company certificate pre-installed, and only need to verify the company signature, reducing energy and network use.

B. Conclusion

It is clear that in order for Wireless Sensor Networks to outgrow their current static use cases, new management and security mechanisms have to be designed. Accepting that sensor networks are not static nor owned by a single company but are mobile and federated environments poses some interesting new challenges on top of the traditional problems of scalability and resource limitations.

At this time little research has been performed to securely deploy keys in Mobile Federated Wireless Sensor Networks. Most key agreement protocols assume a number of preloaded secrets which is not feasible in a mobile and federated environment. In the research of vehicle area network security some solutions have been proposed, yet these solutions are too resource intensive to be instantiated on top of sensor nodes.

In this paper we presented a novel symmetrical key deployment protocol for MAnagement of Secret keYs (MASY) which allows sensor nodes to receive a symmetric key from a previously unknown gateway. MASY employs the available back-end infrastructure to set up a trust relationship between the gateway and the company that owns the sensor nodes. This trust relation is used to securely deploy the symmetric key to the sensor nodes. The key can then be used to secure the communication of the sensor node with the gateway, a group key to secure all network traffic, or a master key to generate further keys, depending on the needs of the application.

We implemented MASY on top of the resource constrained Tmote Sky sensor nodes and the resource rich SunSPOT nodes. The MASY implementations show that new key material can be securely deployed to the sensor nodes with a very limited code size and message overhead.

VII. Acknowledgements

This research is partially funded by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy, IBBT, Research Foundation Flanders (FWO) and the Research Fund K.U. Leuven.

References

[1] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM, 2002, pp. 88–97.

[2] R. Jedermann, C. Behrens, D. Westphal, and W. Lang, "Applying autonomous sensor systems in logistics combining sensor networks, rfids and software agents," in *EUROSENSORS XIX conference*, 2005.

[3] B. P. L. Lo, S. Thiemjarus, R. King, and G. zhong Yang, "Body sensor network - a wireless sensor platform for pervasive healthcare monitoring," in *Adjunct Proceedings of the 3rd International conference on Pervasive Computing (PERVASIVE'05*, 2005, pp. 77–80.

[4] C. Huygens and W. Joosen, "Federated and shared use of sensor networks through security middleware," in *ITNG '09: Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1005–1011.

[5] S. A. Camtepe and B. Yener, "Key distribution mechanisms for wireless sensor networks: a survey," Tech. Rep., 2005.

[6] S. Zhu, S. Setia, and S. Jajodia, "Leap: efficient security mechanisms for large-scale distributed sensor networks," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2003, pp. 62–72.

[7] S. Tripathy, "Effective pair-wise key establishment scheme for wireless sensor networks," in *SIN '09: Proceedings of the 2nd international conference on Security of information and networks*. New York, NY, USA: ACM, 2009, pp. 158–163.

[8] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2002, pp. 41–47.

[9] D. F. Aranha, J. Lpez, L. B. Oliveira, and R. Dahab, "Efficient implementation of elliptic curves on sensor nodes," in *Conference on Hyperelliptic curves, discrete Logarithms, Encryption, etc., Frutillar, Chile*, 2009.

[10] U. Roedig and C. J. Sreenan, Eds., *Wireless Sensor Networks, 6th European Conference, EWSN 2009, Cork, Ireland, February 11-13, 2009. Proceedings*, ser. Lecture Notes in Computer Science, vol. 5432. Springer, 2009.

[11] A. Studer, E. Shi, F. Bai, and A. Perrig, "Tacking together efficient authentication, revocation, and privacy in vanets," in *SECON'09: Proceedings of the 6th Annual IEEE communications society conference on Sensor, Mesh and Ad Hoc Communications and Networks*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 484–492.